Monte-Carlo

Seach, Trees, etc

Monte-Carlo

Idea

When a problem is too hard to solve exactly, simulate it randomly thousands of times and observe the outcomes.

Classic example: estimating π:

- Imagine a square with a circle inside it. You throw darts randomly at the square. Some land inside the circle, some outside. The ratio of darts inside vs total darts approximates π/4.

Darts inside circle / Total darts ≈ π/4

Example Tic-Tac-Toe, mid game

- It's your turn (X). The board looks like this:

```
X | _ | O
---------
_ | X | _
---------
_ | _ | O
```

```
[0,0] | [0,1] | [0,2]
---------------------
[1,0] | [1,1] | [1,2]
---------------------
[2,0] | [2,1] | [2,2]
```

- **100 sims** = 100 complete games played out from that position, with both players making random moves until someone wins or it's a draw.

Sim 1: You play [1,2] → random → random → random → X wins ✓

- Cell [0,1] → 100 sims → 31 wins → 31%
- Cell [1,0] → 100 sims → 28 wins → 28%
- Cell [1,2] → 100 sims → 67 wins → 67% ← pick this
- Cell [2,0] → 100 sims → 44 wins → 44%
- Cell [2,1] → 100 sims → 19 wins → 19%

Example 2 : A completely new game you've never seen

Someone hands you the rules to a brand-new board game. You've never played it. There's no book of strategy, no expert knowledge.

Pure Monte Carlo still works:

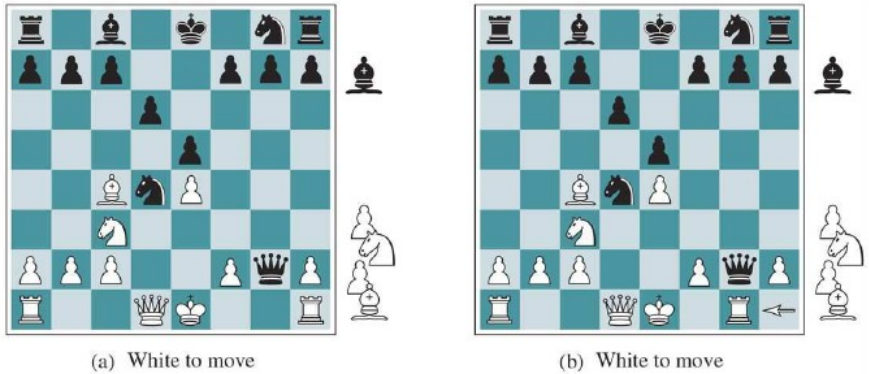
Move A → simulate 100 games → 55 wins

Move B → simulate 100 games → 38 wins

Move C → simulate 100 games → 61 wins ← pick this

You don't need to *understand* the game. You just need to know the rules well enough to simulate it. This is a huge deal, **alpha-beta would be stuck** without a hand-crafted evaluation function.

Where it fails



(a) White to move

(b) White to move

Imagine this position in chess. There's a **one-move checkmate** available. But it's a subtle move, a quiet king step, not an obvious capture.

Move A (checkmate!) → sims end fast, hard to reach by random play → 45%

Move B (aggressive) → random games look exciting but lose late → 52% ← wrongly picked

Move C (defensive) → random games drag on, result unclear → 41%

Random play **never finds the checkmate reliably** because random games don't tend to play the exact sequence that demonstrates it. Pure Monte Carlo misses it.

This is the core weakness: **random playouts are dumb**. They don't know that some moves are critically important.

Monte-Carlo

Works well

New/unknown games

High branching factor

No heuristic needed

More time = better answer

Struggles

Decisive single moves

Subtle tactical sequences

When random play is misleading

Wastes sims on bad moves equally

wastes sims on bad moves equally, this is exactly the problem Monte-Carlo Tree Search (MCST) fixes.

Idea of MCST

Problem: Pure Monte Carlo wastes equal simulations on all moves, even obviously bad ones.

MCTS fixes this by building a tree as it goes, remembering what it learned, and spending more simulations on promising moves.

Core idea MCST

Instead of blindly giving 100 sims to every move, MCTS keeps a scoreboard that updates after every single simulation:

Given A, B, and C are the **3 possible moves** available to you at that moment in the game.

A (wins / total simulations played)

A(1/1) = Move A was simulated 1 time, won 1 time

A(3/5) = Move A was simulated 5 times, won 3 times

A(0/4) = Move A was simulated 4 times, won 0 times

Notation

After sim 1: A(1/1) B(0/0) C(0/0) → tried A, won

After sim 2: A(1/1) B(0/1) C(0/0) → tried B, lost

After sim 3: A(1/1) B(0/1) C(1/1) → tried C, won

A(1/1) → Move A was tried, won → 1 win out of 1 sim

B(0/0) → Move B not tried yet → 0 wins out of 0 sims

C(0/0) → Move C not tried yet → 0 wins out of 0 sims